Noname manuscript No.
(will be inserted by the editor)

High-quality tree structures modelling using local convolution surface approximation

Xiaoqiang Zhu · Xiaogang Jin · Lihua You

Received: date / Accepted: date

Abstract In this paper, we propose a local convolution surface approximation approach for quickly modelling tree structures with pleasing visual effect. Using our proposed local convolution surface approximation, we present a tree modelling scheme to create the structure of a tree with a single high-quality quad-only mesh. Through combining the strengths of the convolution surfaces, subdivision surfaces and GPU, our tree modelling approach achieves high efficiency and good mesh quality. With our method, we first extract the line skeletons of given tree models by contracting the meshes with the Laplace operator. Then we approximate the original tree mesh with a convolution surface based on the extracted skeletons. Next, we tessellate the tree trunks represented by convolution surfaces into quad-only subdivision surfaces with good edge flow along the skeletal directions. We implement the most time-consuming subdivision and convolution approximation on the GPU with CUDA, and demonstrate applications of our proposed approach in branch editing and tree composition.

Xiaoqiang Zhu
State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, P.R. China
E-mail: zhuxiaoqiang@zjucadcg.cn

Xiaogang Jin
State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, P.R. China
Tel.: +86-571-88206681 ext 507
Fax: +86-571-88206680
E-mail: jin@cad.zju.edu.cn

Lihua You
E-mail: LYou@bournemouth.ac.uk
National Center for Computer Animation, Bournemouth University, Bournemouth, UK

Keywords Tree modelling · Convolution surfaces · CUDA · Subdivision surfaces · Quadrilateral meshes

1 Introduction

Trees are ubiquitous natural objects. Their modelling is an essential element in virtual environments. This raises the problem of how to model and edit various trees conveniently, realistically, and quickly. Tree models can be reconstructed from real trees using laser scanners [57, 31] or computer vision techniques [52, 51]. As trees exhibit a large amount of self-similarity, more new tree models can be produced by making full use of acquired tree models. A challenging problem is how to find a new compact representation of tree models with smooth branch ramification so that modelling and editing new trees from existing ones can be performed easily and interactively. Such a modelling tool is also required to simulate plant pruning and grafting, which are important in virtual agriculture to avoid the physical process.

A tree branch usually has a cylinder-like appearance, and it can be abstracted as line segments. Moreover, trees have smooth branches. Based on these observations, in this paper, we propose a convolution surface-based tree representation for efficient skeleton-based tree modelling and editing. Our goal is to develop an interactive skeleton-driven tree modelling approach that can provide easy and high level editing using the skeleton-based local convolution surface approximation and compact quad-only mesh representation. With the method, users can manipulate low-dimensional skeletons to create new trees, and smooth branches are automatically achieved during the editing process. In addition, it can output compact quad-only meshes with good edge flows by making use of the fact that the area of cylinder-like branches occupies a large portion of the whole tree trunk.

We note that smooth branch ramification is not considered in most of previous approaches [57,31,52, 51]. Although implicit surfaces, especially convolution surfaces, can be introduced to create smooth branching structures [9,21,25], the Marching cubes polygonizations of the iso-surface they employed suffers from high computation complexity, limited resolution, and low-quality triangle meshes. Furthermore, it is prone to missing small twigs for complex tree models because the output of the Marching cubes is resolution-dependent. Even though there are a large number of improvements [55,8,10,1,40,59], it is still difficult to balance the quality of the iso-surface polygons and the performance. To solve these problems, we propose an interactive GPU-based quad-only tessellation method to polygonize convolution surfaces with good edge flows along the skeletons of tree models.

Our paper has the following technical contributions: (1) a novel GPU-based local approximation method to represent a given tree model with convolution surfaces, (2) an interactive tree modelling system which combines the strengths of the skeleton-based composition, convolution surfaces and GPU to achieve excellent performance in tree modelling with high-quality meshes.

The remainder of the paper is organized as follows. After introducing the related work in Sect. 2, the generation of the mesh topology and the fitting schemes are presented in Sect. 3 and Sect. 4 respectively. Then the implementation details are presented in Sect. 5 followed by some applications in Sect. 6, and our paper ends with the conclusion section.

2 Related work

Besides trees modelling techniques, our work also involves subdivision surfaces and skeleton-based convolution modelling. In this section, we review the relevant research work.

Modelling trees Tree modelling can be classified into two categories: designing virtual trees [16,38,39] and reconstructing real trees [31,57]. The grammar-based procedural modelling [16], sketch-based modelling [39] and most of the image-based modelling approaches [38] fall into the first category. Early L-systems [30] generate trees from a given initial state, and the recent grammar-based tree modelling [50] designs an algorithm to control the process at a higher level. Although technical users are capable of modelling many excellent distinct trees, rules are too abstract for novice users. Therefore, interactive sketch-based interfaces have been developed to generate 3D trees from sketched 2D shapes [15, 39,41,54,33] during the past decades. The *TreeSketch* system [33] is a popular sketch-based iPad app. for producing trees interactively. Another important approach for modelling virtual trees is image-based modelling. Reche-Martinez *et al.* [44] propose a purely image-based modelling method. The approach proposed in [38] combines image-based with sketch-based modelling to produce 3D tree models from several images. User interactions are also allowed in image-based tree modelling presented by Tan *el al.* [51,52]. With the development of scanning technology, reconstructing real trees from laser point clouds has been developed recently. The tree skeletons are extracted from point sets and then leaves are randomly added to the branches in [57]. Bucksch *et al.* [11] partition points into clusters before connecting adjacent clusters to generate the skeletons. Pirk *et al.* [43] simulate the natural growth of trees and interaction with their environment. The component-based tree synthesis is discussed and mesh fusion is adopted in [36], it works on 3D meshes directly, and the connections between different parts are allowed for branches with predefined similar contours in 3D meshes. In [32], Lluch *et al.* propose a new scheme for producing a single polygonal mesh by refining the junctions. Lin *et al.* [29] create branching shapes by fusing disconnected mesh components. Galbraith *et al.* [19] present an implicit surface-based tree modeling which can simulate bud scale scars and branch bark ridges, and the ray-tracing method and the polygonization are both used to render the result models.

Subdivision surfaces Some of commonly used subdivision surfaces such as Catmull-Clark [14], Loop [34], Doo-Sabin [17] and $\sqrt{3}$ [28] subdivisions have been integrated in the CGAL (Computational Geometry Algorithm Library [18]) and OpenMesh [48], and executed on CPU. Recently, more and more research activities contribute to GPU implementation [12,35, 56] because of GPU's programmability and massive computational capability. With the recent emergence of CUDA (Compute Unified Device Architecture), many researchers have used it to design parallel subdivision algorithms [42,46]. The subdivision scheme with CUDA adopted in this paper is based on the parallel Catmull-Clark subdivision proposed in [42].

Skeleton-based convolution modelling As natural abstracts of shapes, skeletons capture the essential topology of an object in a very compact form, and they can be easily edited [60]. A convolution surface is defined as an iso-surface in a scalar field by convolving a geometric skeleton with a kernel function [9,37]. Analytical solutions are the best way to reduce the considerable calculation for the convolution integrals [22,25–27,37, 47,60,23]. Since convolution surfaces offer a number of advantages such as smoothness, fluidly varying topology, local control, well behaved blends, and simple implementation, they have numerous applications in design, modelling and animation. One interesting application is to model sketch-based models [4,3,2,7,49, 60] which uses the advantages of the rotundity and smoothness of convolution surfaces. Another important application of convolution surfaces is to simulate deformable surfaces with complex topology, and the imbedded skeletons make it especially suitable for skeleton-based animation or special effects. Due to the advantages of skeletal abstraction and varying topology, skeleton-based convolution surfaces have also been used to model tree branches [37] and organic shapes [9] conveniently.

Convolution fitting is an important process in convolution surface-based 3D modelling. Some fitting techniques have been successfully developed in [4,3,2,7]. By using a new bounded function, a fitting technique is proposed in [4] which combines the initial surface approximation with a minimization adjustment procedure to achieve optimal fitting, better smoothness, and less oscillation. The technique presented in [2,3] uses a polynomial function to automatically assign weights of convolution surfaces and avoid any adjustments afterwards. The polynomial function is determined by best interpolating all the values found during manually fitting the weights for a large number of cases. This technique has the benefits of a good approximate fit of the user drawn contour and a smooth non-oscillating surface. In order to avoid the optimisation operation required in a fitting process, the fitting technique described in [7] determines the convolution weights with a third power of the scaled radius and the iso-surface of the field function by averaging the field values at the points located on the contour. For tree modelling, the radius variation between neighboring nodes is small which guarantees tiny oscillations. Therefore, we will not address this issue in this paper. Since efficiency is one of the important factors in tree modelling which involves many skeletons, we will present an efficient local approximation fitting technique in this paper. Such a technique avoids time-consuming solution of the constrained least-squares problem and selection of constraint positions required in the global approximation, and has the benefit of naturally stitching tree branches together. Therefore, it is especially suitable for the tree structure modelling given in this paper.

3 Generation of quad-only mesh topology

3.1 Skeleton-based bounding polyhedron

Our work begins with an existing tree model, which can be created with modelling tools or reconstructed from real data. Then the line skeletons are extracted using the same approach as in [59], where an implicit Laplacian smoothing operator is involved [6,13]. Besides the extracted skeletons, a useful by-product of induced skeleton-mapping is generated. That is, for each skeleton node N_k, the mapping set of vertices $\prod_k$ on the original mesh are contracted and collapsed to N_k. The mapping set of vertices at each node forms a cylinder-like shape (non-branching node) or a sphere-like shape (branching node). For each mapping set $\prod_k$, we calculate its approximate thickness r_k by averaging the distances between N_k and its mapping vertices in $\prod_k$ (Fig. 1). As skeleton nodes with too high density hinder the performance, we resample the skeletons by deleting unnecessary nodes. To do this, we discard the non-branching node N_k if $\angle N_{k-1}N_kN_{k+1} < \varepsilon$ or $\min(|N_{k-1}N_k|, |N_kN_{k+1}|) < cr_k$, where both ε and c are user-specified parameters.

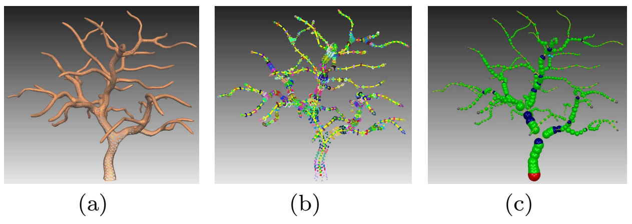


Fig. 1 An input acer japonicum model (a), its induced skeleton-mapping (b) and the thickness of each mapping set (c)

After that, the bounding polyhedron is created based on the skeletons on CPU, which consists of mainly quadrilaterals and a few triangles (Fig. 2). The details can be found in Sect. 4.2 of [24]. The created polyhedrons are used as a control mesh for the following subdivision.

3.2 Parallel quad-only surface subdivision

Quadrilateral meshes are more preferable for artists because of their natural characteristics. Once the graph-based tree skeleton has been created, the approach adopted in [24] is employed to create a polyhedron which serves as a control mesh (Fig. 2). Each facet of the initial control mesh has three or four edges. A

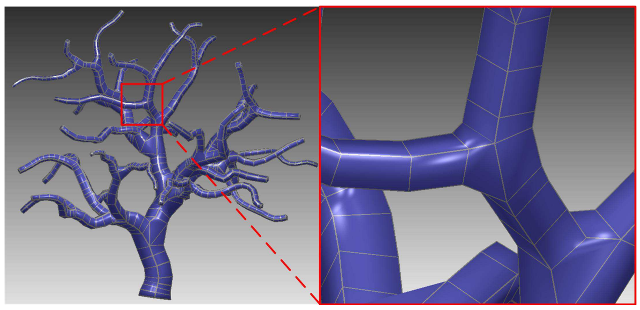


Fig. 2 The created bounding polyhedron of the tree skeletons

quad-only mesh is generated after one or more levels of the Catmull-Clark subdivisions below. Our method has advantages over the previous iso-surface extraction methods such as Marching cubes, which usually miss small features like thin twigs and extract triangles with low quality.

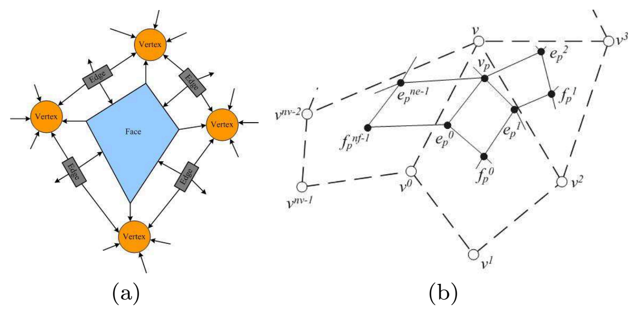


Fig. 3 Topological information in the mesh structure (a) and Catmull-Clark subdivision scheme (b)

Given a control mesh M_0, a new mesh M_1 consisting of a collection of quadrilaterals is produced by applying the Catmull-Clark subdivision schemes to M_0. With the repetition of such a scheme, M_1 is recursively subdivided into M_2, then M_3, and so on. The Catmull-Clark subdivision process has been implemented with the half edge structure on CPU, and even integrated into mesh libraries such as OpenMesh, and CGAL. For GPU implementation, collaborated design is essential. Patney *et al.* [42] present a data structure (Fig. 3 (a)) for Catmull-Clark subdivision surfaces with CUDA on GPU. For each subdivision step, three types of new vertices are calculated to form the subdivision surfaces (Fig. 3(b)). Therefore, three relevant CUDA functions are designed to calculate them in parallel respectively:

- *Facet point* $f_p = \frac{1}{n} \sum_{i=0}^{n} v^i$, where v^i is the i^{th} vertex of the current facet and n is the vertex count of the facet. In this step, one thread is dispatched for each facet to evaluate the centroid of its four corners. After that, the dispatched thread also adds the calculated face point coordinates to the corner vertices of the facet for updating the vertex points.
- *Edge point* $e_p = \frac{1}{4} \left\{ \sum_{i=0}^{1} v^i + \sum_{i=0}^{1} f_p^i \right\}$, where v^i and f_p^i are the i^{th} vertex and the i^{th} adjoining face of the current edge. The relevant CUDA kernel function distributes a thread for each edge to calculate the edge point by averaging four vertices: the two vertices of the current edge, and the two face points of the adjoining facets. Similar to the above-mentioned updates of vertex points, the current thread will perform an extra task, adding twice the coordinates of the edge points to its two vertices.
- *Vertex point* $v_p = \left(\frac{n-2}{n}\right) v + \frac{1}{n^2} \sum_{i=0}^{n-1} v^i + \frac{1}{n^2} \sum_{i=0}^{n-1} f_p^i$, where v and n are the original vertex and its valence (the number of neighbouring vertices), and the same definition of v^i and f_p^i as above is applied here. During the GPU implementation, as the face points and the edge midpoints have already been aggregated into the adjacent vertex points, the following equation is used to evaluate the updated vertex point: $v_p \Leftarrow \frac{v_{in} \times (n-3) + \frac{v_{out}}{n}}{n}$, where v_{out} contains the already aggregated face points and twice the midpoints of adjoining edges, and v_{in} is the old coordinates of the vertex.

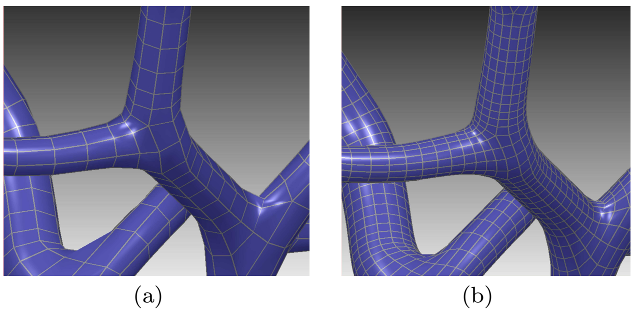
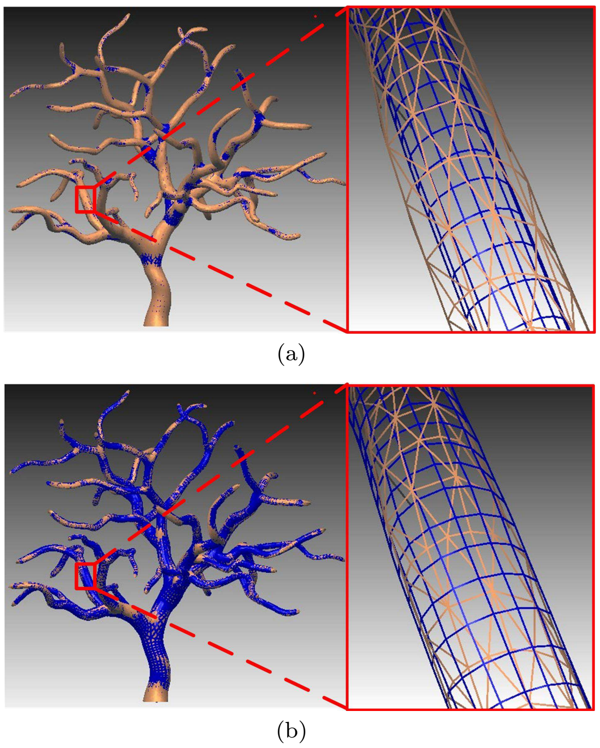


Fig. 4 The results of the created bounding polyhedron shown by the right image of Fig. 2 after the 1st subdivision (a) and the 2nd subdivision (b), respectively

As described in [42], here we also maintain two copies of vertices, edges and faces information in order to produce new data from the old copies, and swap the pointers to the old and the new copies after each subdivision. With this update scheme, a smooth and shrunk mesh is generated after several steps. The final result is mapped to VBs (vertex buffers) through the CUDA-OpenGL interoperability, so it can be rendered with a single GPU call (Fig. 4).


(a)

(b)

Fig. 5 (a) The overlapped results of the original mesh (orange) and the naive subdivision (blue). (b) The overlapped results of the original mesh (orange) and the locally approximated result with Cauchy kernel (blue)

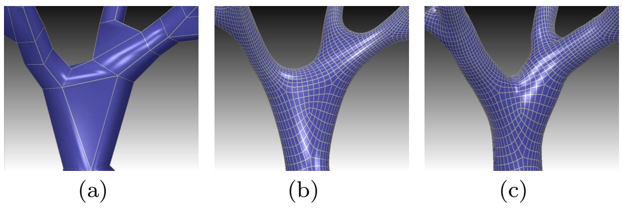

(a) (b) (c)

Fig. 6 The sparsely sampled skeleton nodes give rise to large facets of a control mesh (a) and too flat ramifications after naive subdivisions (b), but the convolution approximation can create rounded ramifications (c)

4 Fitting with convolution surfaces

The superposition and smooth blending properties of convolution surfaces make it extremely suitable for modelling tree-like shapes. Although the excellent mesh topology after the subdivision can be produced, the newly generated vertices are not on the skeleton-based convolution surfaces due to the above-mentioned shrinkage (Fig. 5 (a)). In addition, the vertices at ramifications are seriously dependent on the control vertices (Fig. 6 (b)), which can be solved through the superposition of convolution surfaces (Fig. 6 (c)). In order to address this issue, we perform an evolution on the subdivision surfaces to make them better approximate the target convolution surfaces (Fig. 5 (b)).

The result of the above-described subdivision is a topology of quad-only mesh which will be approximated to form an excellent shape. The target limit surface is defined as a convolution surface based on the embedded line skeletons.

4.1 Convolution approximation

The convolution surface S based on a series of skeletons can be defined in the general form below:

$$S = \left\{ \mathbf{p} | \sum_{i=1}^{n} \lambda_i F_i (\mathbf{p}) - T = 0 \right\}, \tag{1}$$

where F_i and λ_i are the field function and the field contribution weight of the i^{th} skeleton segment respectively, and T represents the threshold value of the iso-surface.

Previously, convolution approximations at given constraint positions $\{p_1, ..., p_m\}$ are commonly performed by solving a constrained least-squares problem below for $\Lambda = [\lambda_1, ..., \lambda_n]^\top$ [49]:

$$\min_{\Lambda \geq 0} (\mathbf{F}\Lambda - \mathbf{T})^\top (\mathbf{F}\Lambda - \mathbf{T}), \tag{2}$$

Such convolution approximations have two problems. First, the iterative method or NNLS (Non-Negative Least Square) used to solve the constrained least-squares problem is rather time-consuming especially when there are too many skeleton segments and constraint points. Second, special care must be taken to choose the constraint positions. Despite this, the resulting global approximations are still not satisfactory especially at branch nodes as shown in Fig. 7.

4.2 Analytical solutions for local approximation

In order to address the above issues, we present a local approximation solution here, which is especially more suitable for our modelling tree purpose because it has the following advantages:

- Since the local approximation instead of the global one is used, no least-squares problem is involved. Therefore, our local approximation is more efficient.
- Tree branches are naturally stitched because of the superposition property of convolution surfaces, which guarantees the smooth blending, and avoids the operation of positioning constraint points at branches used in the global approximation.

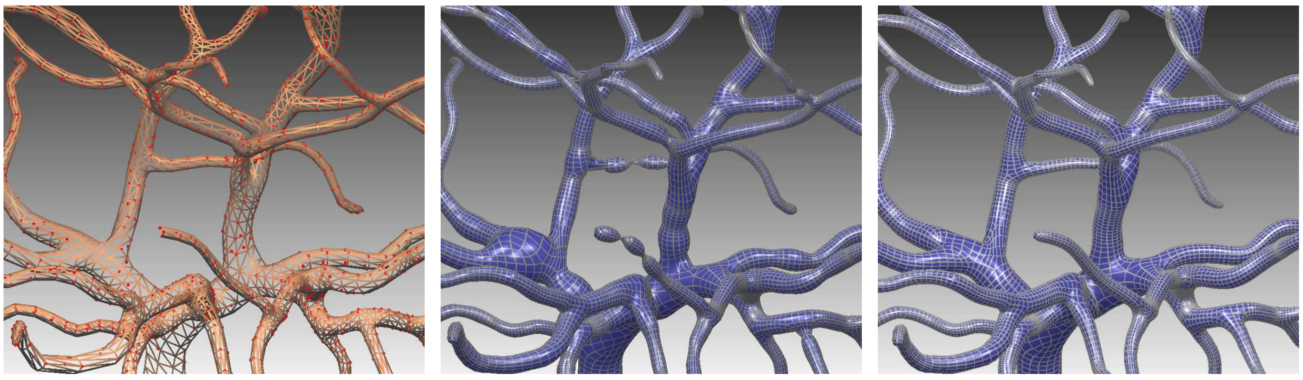


Fig. 7 The close-ups of the chosen 1548 constraint positions ((a), red points) for the global approximation ((b), 520 skeletal segments). The image in (c) illustrates our local approximation with Cauchy kernel

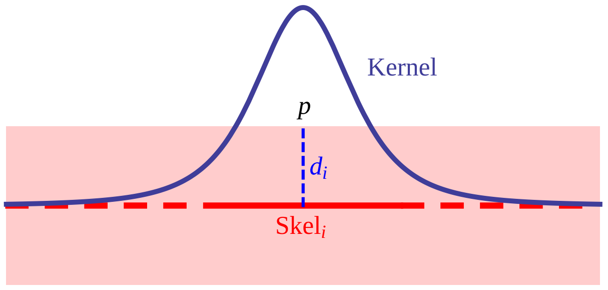


Fig. 8 The sketch of the local fitting

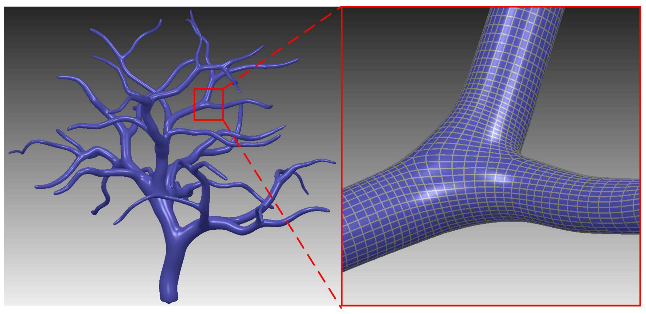
The basic idea of our local approximation is to treat each skeleton segment as a line with infinite length (Dashed lines in Fig. 8 are imaginary skeleton segments), and determine the weight of its field contribution by both the iso-value and the distance between the current segment and the original mesh. For a skeleton segment $Skel_i$ with the radii r_a and r_b at its two ends, a point $\mathbf{p}$ at $d_i = \frac{r_a+r_b}{2}$ distance from $Skel_i$ (Fig. 8) needs to be fitted. Therefore, the weight of $Skel_i$ can be determined as:

$$F_{Cauchy}(p) = 2\lambda_i \int_0^\infty \frac{1}{\left(1+s^2\left(x^2+d_i^2\right)\right)^2} dx = T \\ \Rightarrow \lambda_i = \frac{2}{\pi} sT \left(1+s^2 d_i^2\right)^{\frac{3}{2}} \tag{3}$$

$$F_{Quartic}(p) = 2\lambda_i \int_0^{\sqrt{R_i^2-d_i^2}} \left(1-\frac{d_i^2+x^2}{R_i^2}\right)^2 dx = T \\ \Rightarrow \lambda_i = \frac{15TR_i^4}{16\left(R_i^2-d_i^2\right)^{\frac{5}{2}}} \tag{4}$$

where R_i is the effective radius of the kernel for the current skeleton segment which is empirically set to $2d_i$ in practice, and λ_i is the final weight of the skeleton segment.

The solutions for these two kernels are both based on the assumption that the kernel functions are infinitesimal beyond some distance, which can be satisfied by choosing a suitable control parameter s for the Cauchy kernel and effective radius R for the quartic polynomial kernel. The excellent results from the above-described kernels are obtained and illustrated in Fig. 9. Observing the images given in the figure, we find an impressive similarity between the results obtained from the two different kernels. The small radius variation between neighboring nodes guarantees tiny oscillations of the final convolution surfaces. Due to the effective local support, the quartic kernel consumes less time


(a)

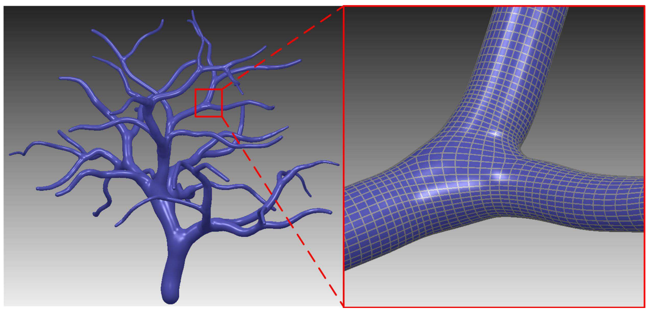

(b)

Fig. 9 The results after the 2*nd* subdivision and the fitting with two different kernels: Cauchy kernel (a) and quartic kernel (b)

(about 1.5 times faster in our experiments). In order to meet different user preferences, both kernels are incorporated in our system. For the sake of conciseness, we take the Cauchy kernel as an example in this paper if no specification is involved.

4.3 Surface evolution

The approximation is obtained by moving the vertices of subdivided surfaces to the convolution surfaces in the gradient directions that continuously change in the whole 3D domain, and the standard Newton iteration can be performed to project the vertex onto the convolution surface [53]:

$$v_i \leftarrow v_i + \sigma \left(F\left(v_i\right) - iso\right) \frac{\nabla F\left(v_i\right)}{\left\|\nabla F\left(v_i\right)\right\|^2}, \quad (5)$$

where $\sigma = 0.95$ is an effective value for our application. In the parallel evolving implementation, one thread is dispatched for each vertex. In our experiments, an average of 5 to 7 evolving steps for each vertex is sufficient for all the examples in this paper.

4.4 Error reduction

There are errors in our approximate process, which can be seen in Fig. 10 (a). In order to intuitively visualize the approximate errors, we map it to a colour space, where positive errors mean that the approximated vertices fall outside of the original mesh, and vice versa. In order to further reduce the approximate errors, our system presents an optional off-line function to restore the offsets which are defined as the vectors from the approximated vertices to the original mesh in the normal directions. The resulting model after restoring the offsets is shown in Fig. 10(b) which clearly demonstrates a significant reduction of the approximate errors.

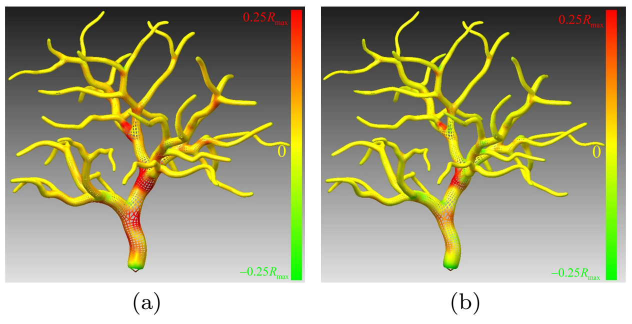


Fig. 10 (a) The approximate errors. (b) The errors after restoring the offsets

5 Results and discussions

Our system is tested on a PC equipped with Intel Q9550 CPU @2.83GHz with 4GB of memory, and the GPU of Nvidia GeForce GTX 280 with an 1GB of dedicated memory. The most time-consuming subdivision and convolution approximation are both performed on GPU. At each subdivision, the surface data of the previous subdivision is stored in graphics memory as textures, which can accelerate the memory access due to the cache mechanism. Another cache mechanism of CUDA is for constant memory, which is used to store the numbers of vertices, edges and faces for efficient access of each thread. As the numbers of vertices and faces at each subdivision can be pre-computed, the allocations of graphics memory are all carried out only once.

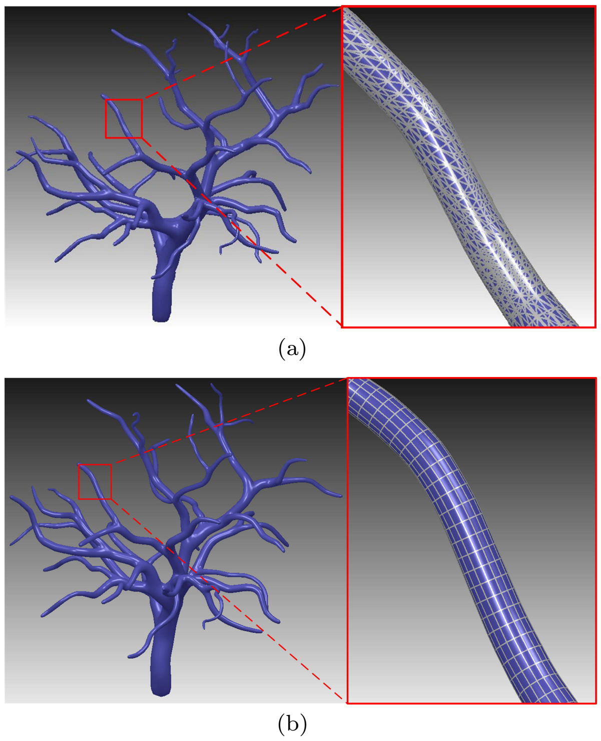


Fig. 11 The produced polygonal iso-surface with improved Marching tetrahedra (a) and our scheme (b). All the results are based on the local approximation with Cauchy kernel

In order to extract the iso-surface of the tree trunk-based convolution surface, the improved Marching tetrahedra [59] (Fig. 11(a)) produces triangles efficiently, which has better visual effects and achieves higher performance than the traditional Marching tetrahedra. Compared to the improved Marching tetrahedra, the subdivision scheme adopted in this paper

Table 1 Improved Marching tetrahedra. Abbreviations: (AJ) Acer japonicum, Skels (Skeletons), Tets (Tetrahedrons), TetVerts (Tetrahedral Vertices), IsoTris (Iso-surface Triangles), Plyh (Polyhedron Generation), Tetz (Tetrahedralization), TetSub (Tetrahedron Subdivision), FieldCalc (Field Calculation), IsoExtr (Iso-surface Extraction)

Model	Skels	Tets	TetVerts	IsoTris	Time(s)					
					Plyh	Tetz	TetSub	FieldCalc	IsoExtr	Total
AJ	288	1,180,160	1,010,845	388,951	0.015	0.063	0.065	0.344	0.011	0.498
Ulmus	290	1,102,336	944,239	369,690	0.014	0.062	0.068	0.316	0.01	0.47
Quercus	129	464,384	397,792	161,540	0.008	0.032	0.042	0.07	0.005	0.157

Table 2 Our method. Abbreviations: (AJ) Acer japonicum, Skels (Skeletons), IsoQuads (Iso-surface Quadrilaterals), Plyh (Polyhedron Generation), SurfSubd (Surface Subdivision), Appr (Approximation)

Model	Skels	IsoQuads	Time(s)			
			Plyh	SurfSubd	Appr	Total
AJ	288	17,088	0.015	0.016	0.142	0.173
Ulmus	290	17,160	0.014	0.018	0.154	0.186
Quercus	129	7,704	0.008	0.013	0.031	0.052

Table 3 Time of solving field weights of skeleton segments using local/global approximation

Approximation	Figures	Constraint Points	Time (s)
Local	Fig. 7(c)	/	**0.006**
Global	Fig. 7(b)	1,548	2.038
	Fig. 12(a)-(b)	1,548	2.025
	Fig. 12(c)-(d)	532	1.105
	Fig. 12(e)-(f)	4,999	5.183
	Fig. 12(g)-(h)	5,516	6.002

makes sure that all the quadrilaterals are uniformly distributed all over the whole tree (Fig. 11(b)) and produces far fewer polygons for the same rendering quality.

Moreover, our subdivision scheme is more efficient. This is because the expensive calculation of field values for a convolution surface usually becomes a bottleneck in the whole process, and the number of positions in Marching tetrahedra for field calculation is tens of times larger than that of our approach. Even though our field calculation at each vertex is performed several times during the approximation stage, the computational cost is still small (Time\Appr in Table 2) compared with the Marching tetrahedra (Time\FieldCalc in Table 1). In addition, our method is free from the *locality problem* [5,20] over the traditional iso-surface extraction approaches since the potential field values of the vertices in our quadrilateral mesh can be calculated using nearby skeletons according to the structural relationship between the initial control mesh and the skeletal segments. The more advanced technique dealing with the locality problem has been successfully developed in [5].

Compared to the global approximation, our local approximation is free of serious bugles (Fig. 7). If the global approximation is employed, the constraint points have to be carefully selected, as the global approximation is extremely dependent on both the number of constraint points and their distribution. Worse results will occur when the selected constraint points are non-uniformly distributed all over the tree (Fig. 12(a)-(b)) or too few constraint points are selected (Fig. 12(c)-(d)). In our experiment, even though all the vertices of the original mesh are taken as constraints and they are uniformly distributed, a rugged surface can still be obviously seen (Fig. 12(e)-(f)). Both additional auxiliary points and a post-smoothness have to be involved to produce a more pleasing result (Fig. 12(g)-(h)). In contrast, our local approximation is independent of constraint points, the only essential information is the radius of each skeletal node that can be easy to obtain, and further smoothed by simply averaging the radii of the neighboring nodes of the current node. On the other hand, to solve the weight of each skeleton segment (Table 3), the adopted NNLS in global approximation has to solve an unconstrained least squares problem in every iteration. Therefore, with the constraints increasing (Constraint Points in Table 3), more and more time will be spent on solving the NNLS system. In comparison, the local approximation does not suffer from this problem, as the weight of each skeleton segment is calculated separately using a pre-defined analytical formula. Another advantage is the locality. That is to say, only the weights of the current parts are to be recomputed and updated when a new branch is added or deleted. For the global approximation, adding or deleting a new branch will result in a completely new NNLS system which has to be solved to determine all the weights.

When a branch has too many child branches as illustrated in Fig. 13, our method will however result in large errors because the original mesh at the branch is too dissimilar to a cylinder-like shape. In spite of this, our method still produces more natural surfaces than the global approximation.

If we use the B-Mesh method [24] to model tree trunks, many metaballs have to be employed to generate a smooth surface for a long straight branch. With our method, only one line segment is sufficient. Due to

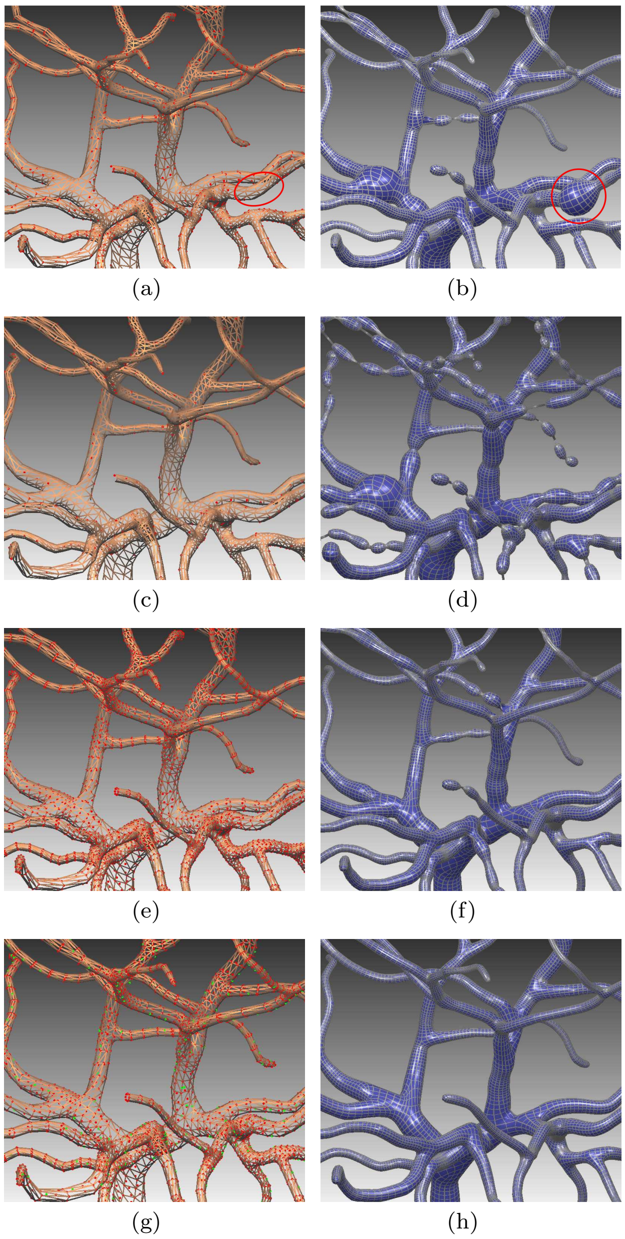


Fig. 12 The close-ups of the global approximation using different numbers of constraint points. (a)-(b): the same number(1,548) of constraint points as used in Fig. 7, but here they are non-uniformly distributed across the entire tree model. (c)-(d): The uniformly selected 552 constraint points and the approximate result. (e)-(f): All the 4,999 vertices of the original mesh are adopted. (g)-(h): All the 4,999 vertices of the original mesh and additional 517 auxiliary mid-edge points (green points) are adopted, then the result weight is post-processed by averaging adjacent weights of each skeleton

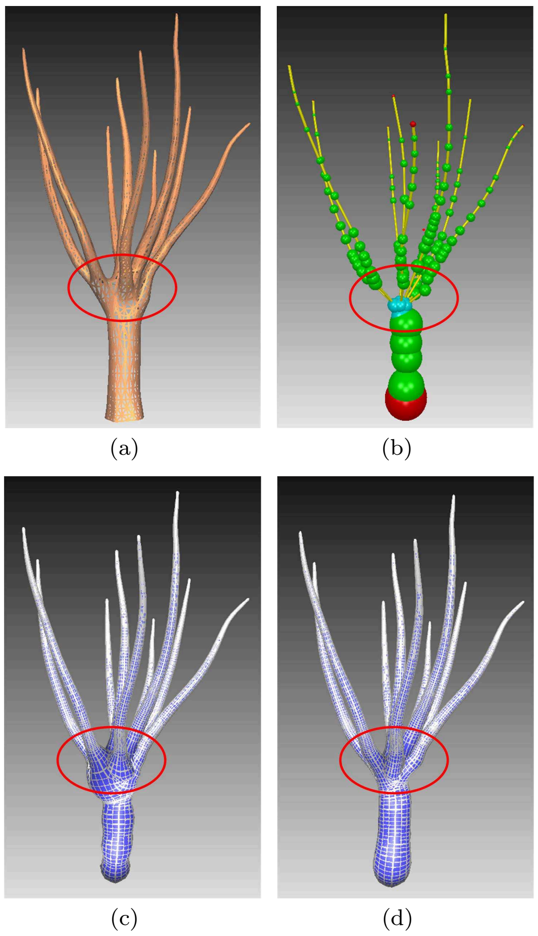


Fig. 13 An example with too many child branches produces large errors. (a) The original mesh; (b) the extracted skeleton and its thickness information; (c) the globally approximated results; (d) the locally approximated results

this reason, our algorithm is far more efficient than the B-Mesh approach. Comparing the GPU implementation of our algorithm with the CPU implementation of the B-Mesh approach, our method is more than fifteen times faster for all the examples presented in this paper. In comparison with the GPU implementation of the B-Mesh approach, the GPU implementation of our algorithm is still about two times faster (the speedup is 2.10 for Acer japonicum, 3.49 for Ulmus, and 4.07 for Quercus).

Although our scheme is derived from the B-Mesh [24], two great improvements have been achieved to make our scheme more suitable for modelling tree models. The first improvement is the adoption of line skeleton-based convolution surfaces rather than point skeleton-based metaball surfaces. Such a treatment solves the following problems of adopting point-based metaball surfaces: a) determining how many metaballs should be placed between each pair of skeleton nodes is avoided since only a line skeleton for convolution surfaces is used in our scheme, b) when metaball surfaces are used to approximate the tree models, bulges will occur if insufficient metaballs are provided or the metaballs are not placed elaborately, especially at branching nodes.

Although the bulges can be reduced to some extent when sufficient metaballs are provided, too many metaballs will result in a heavy computational burden in the approximation step. By contrast, our line skeleton-based convolution surfaces have no bulge artifacts. The second improvement is the parallel computing of the Catmull-Clark subdivision and convolution approximation, which are the most computation-intensive. With our parallel computing implementation, an interactive performance is achieved, even when many skeleton segments (500 skeleton segments in our experiment) of tree models are involved.

To further demonstrate the robustness of our approach, we present trees with large radius differences in Fig. 14.

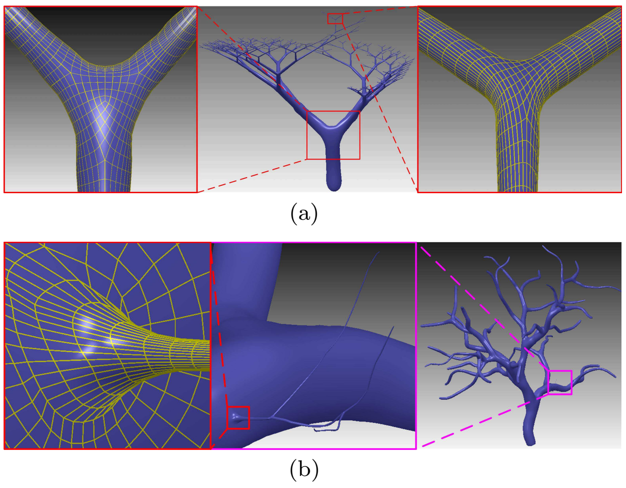


Fig. 14 Trees with large differences of radii. (a) A tree generated with an L-System; (b) a tree composed with our system

6 Applications

In this section, we demonstrate the effectiveness of our method for branch editing and tree composition.

6.1 Branch editing

As an excellent abstract of trees, the line skeleton can be interactively split, rotated, translated and appended. The surface model is updated correspondingly and naturally. A similar interface is borrowed from the MeshMixer system [45]. With our approach, it is convenient to make a manifold surface from separated overlapped branches (Figs. 15-16). In Fig. 17, we give three different frames to demonstrate good mesh quality during editing animation of creating tree structure models.

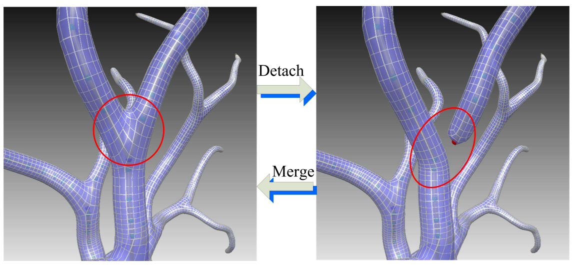


Fig. 15 The detaching and merging procedure

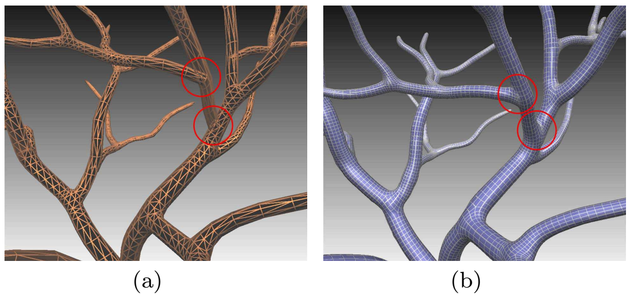


Fig. 16 The original mesh with superimposed branches (a) and our generated manifold shape (b)

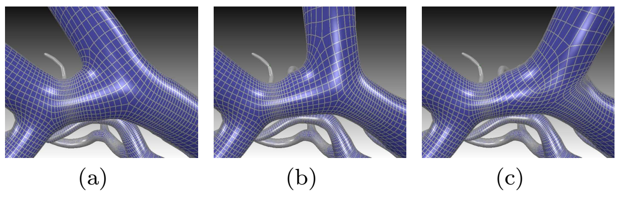


Fig. 17 Different frames during the editing animation. (a) Anticlockwise rotation. (b) Rest pose. (c) Clockwise rotation

6.2 Skeleton-based composition

From an extracted skeleton, we take each child branch of a joint node as a child tree (Fig. 18) and store it in a subtree library. After that, we randomly and recursively replace a child branch with a child tree from the library to generate a series of trees (Fig. 19), which all belong to the same species of the template tree. In the process, we determine the radius of each newly inserted child

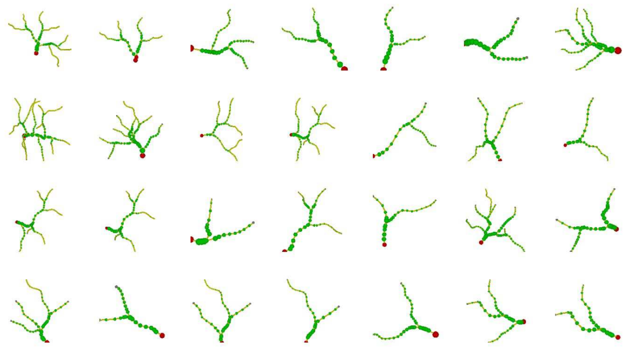


Fig. 18 Child trees generated from the tree given in Fig. 1

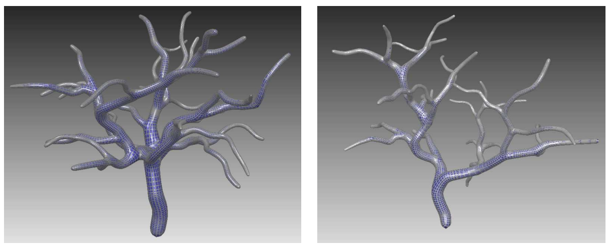


Fig. 19 Two new trees composed with the child trees given in Fig. 18

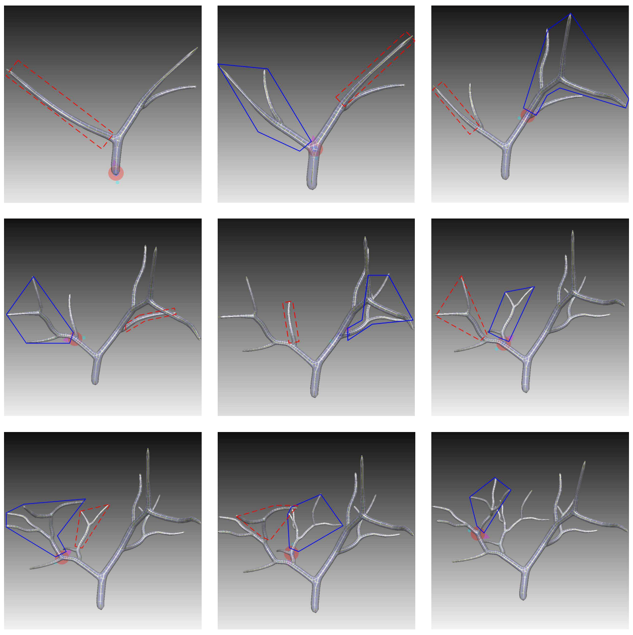


Fig. 20 The composition procedure of a tree model: a branch outlined with the red dashed boundary is to be replaced by the one outlined with the solid blue boundary in the following step

tree according to the proposed knowledge and heuristic rules in [57]. The detailed composition procedure is illustrated in Fig. 20.

With our system, it is easy to model quad-only tree shapes with excellent edge flow from existing tree meshes, and intuitive and efficient to compose new trees by using existing subtrees. In Figs. 21-22, we present more examples of modelling and composition.

7 Conclusions and future work

In this paper, we have proposed a method of modelling high quality quad-only tree shapes efficiently based on the proposed local convolution surface approximation. Our proposed approach has a number of excellent features, indicated below. (1) *Smooth local convolution fitting.* The adopted convolution surfaces based on line skeletons and the local approximation maintain smoothness and natural blending at branches. The superposition and bulge-free properties of a convolution surface allow us to graft arbitrary branches naturally. (2) *Compact representation with high quality meshes.* Compared to iso-surface extraction such as Marching tetrahedra, which generates too many low quality triangles and usually misses small features, our proposed approach has a more compact representation and creates high quality quad-only meshes because: a). they never omit any small features, b). all the meshes have nice edge flows along the line skeletons of tree models, c). the subdivided quadrilaterals are uniformly distributed across the whole tree model, and the edge lengths of the facets are proportional to the radii of the tree branches. (3) *Interactive performance.* As the most time-consuming subdivision and convolution fitting processes are implemented through the parallel computing platform and CUDA architecture, our system is efficient and achieves interactive performance even on a common desktop PC.

The problem of self-intersections has not been tackled in this paper. The advanced technique developed in [5] can be introduced to solve this problem. It will be our future work. In addition, more knowledge and heuristic-based modelling techniques [57,43] will be used to guide the modelling procedure. Another efficient subdivision method is to adopt the hardware tessellation provided only by the most recent graphics cards, which will be used in future experiments to find a more suitable way to model quad-only tree models. Compared to convolution surfaces, homothetic convolution surfaces developed in [58] have advantages of radius control, non-blurring and non-vanishing details and scale-invariant blending. These advantages can be introduced to improve the work given in this paper. Especially, the advantage of non-blurring and non-vanishing details can be employed to deal with the large difference in radii better. We intend to use homothetic

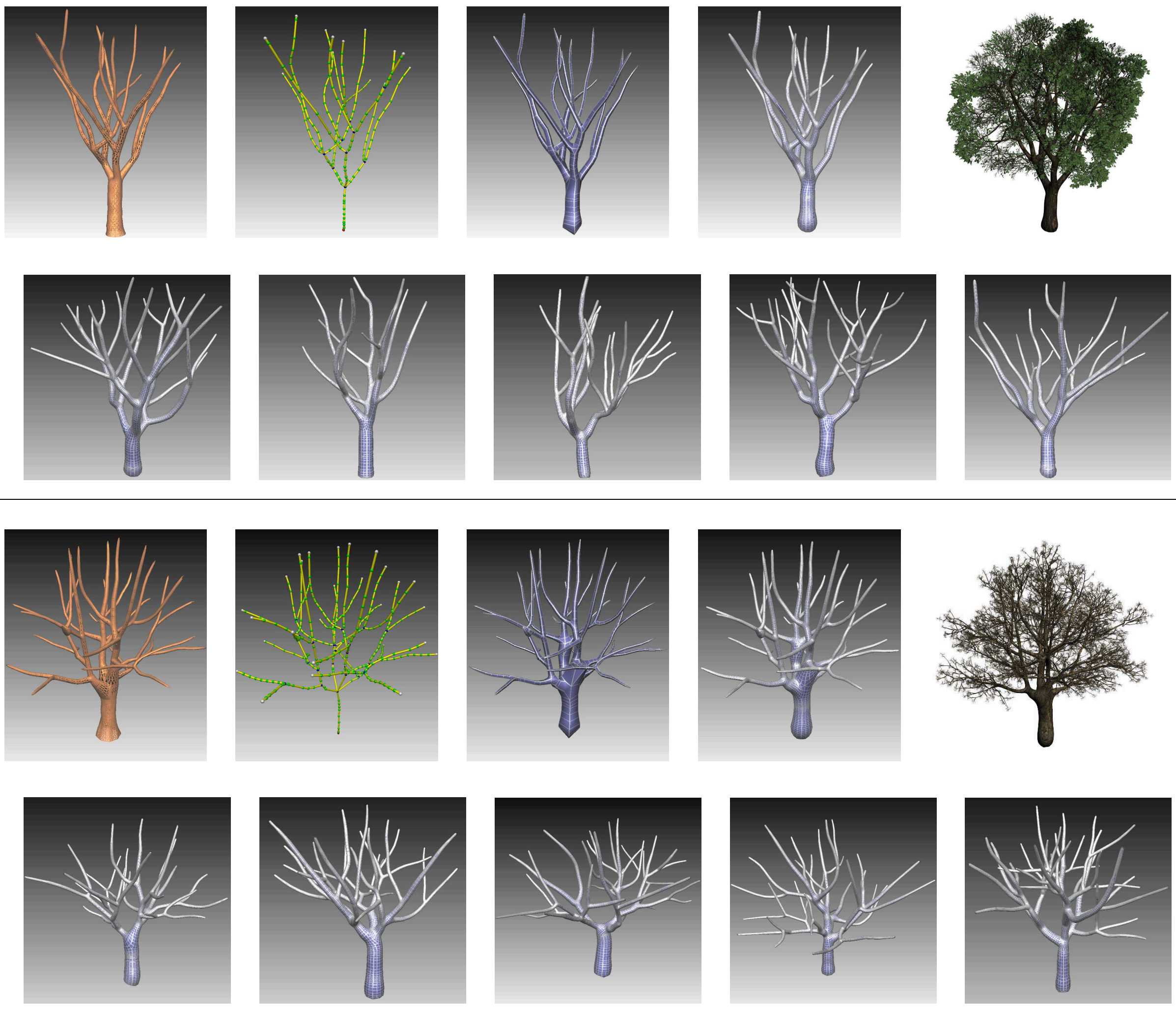


Fig. 21 Two other examples: ulmus and quercus. In the first row of each group, the images show the original tree meshes (the 1*st* column), the extracted line skeletons (the 2*nd* column), the created bounding polyhedrons (the 3*rd* column), the results after the second subdivision and approximation (the 4*th* column), and the final rendering images with texture mapping (some leaves and branchlets are additionally involved) (the 5*th* column). The images in the 2*nd* row of each group are the newly composed trees with the subtrees generated from the one in the 1*st* row

convolution surfaces for tree structure modelling in our future work.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 61272298 and 61373084), Zhejiang Provincial Natural Science Foundation of China (Grant No. Z1110154), the China 863 program (Grant Nos. 2012AA011503 and 2013AA01A603), and the Major Science and Technology Innovation Team (Grant no. 2010R50040).

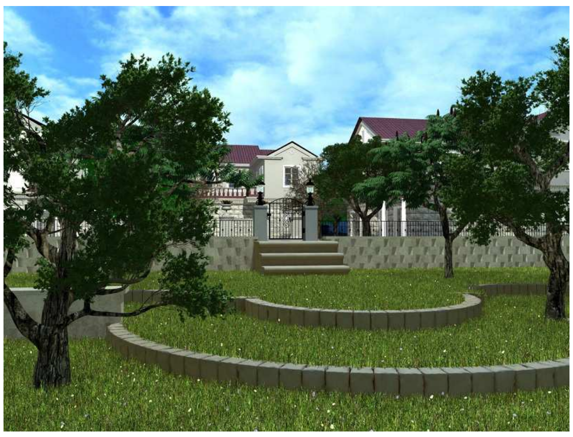


Fig. 22 The modeled quad-only trees in a living quarter

References

1. Akkouche, S., Galin, E.: Adaptive implicit surface polygonization using marching triangles. Comput. Graph. Forum **20**(2), 67–80 (2001)

2. Alexe, A., Barthe, L., Cani, M.P., Gaildrat, V.: Shape modelling by sketching using convolution surfaces. In: In Pacific Graphics (Short Papers) (2005)
3. Alexe, A., Barthe, L., Gaildrat, V., Cani, M.: A sketch-based modelling system using convolution surfaces. In: Technical Report IRIT - 2005-17-R
4. Alexe, A., Gaildrat, V., Barthe, L.: Interactive modelling from sketches using spherical implicit functions. In: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa, AFRIGRAPH '04, pp. 25–34. ACM, New York, NY, USA (2004)
5. Angelidis, A., Jepp, P., Cani, M.P.: Implicit modelling with skeleton curves: Controlled blending in contact situations. In: Proceedings of the Shape Modeling International 2002 (SMI'02), SMI '02, pp. 137–144. IEEE Computer Society, Washington, DC, USA (2002)
6. Au, O., Tai, C., Chu, H., Cohen-Or, D., Lee, T.: Skeleton extraction by mesh contraction. In: ACM SIGGRAPH 2008, SIGGRAPH'08, pp. 44:1–44:10. ACM, New York, NY, USA (2008)
7. Bernhardt, A., Pihuit, A., Cani, M.P., Barthe, L.: Matisse: painting 2d regions for modeling free-form shapes. In: EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, SBIM'08, pp. 57–64. Eurographics Association, Annecy, France (2008)
8. Bloomenthal, J.: An implicit surface polygonizer. Graphics Gems IV pp. 324–349 (1994)
9. Bloomenthal, J., Shoemake, K.: Convolution surfaces. Comput. Graph. **25**(4), 251–256 (1991)
10. Bottino, A., Nuij, W., Overveld, K.v.: How to shrinkwrap a sadle-point: an algorithm for the adaptive triangulation of iso-surfaces with arbitrary topology. Proceedings Eindhoven Implicit Surfaces Workshop (1996)
11. Bucksch, A., Lindenbergh, R., Menenti, M.: Skeltre - fast skeletonization for imperfect point cloud data of botanic trees. In: EG Workshop on 3D Object Retrieval, pp. 13–27 (2009)
12. Bunnell, M.: GPU Gems 2. ch. Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping. Addison Wesley Professional, Boston, USA (2005)
13. Cao, J., Tagliasacchi, A., Olson, M., Zhang, H., Su, Z.: Point cloud skeletons via laplacian-based contraction. In: Proc. of IEEE Conf. on Shape Modeling and Applications, SMI'10, pp. 187–197. IEEE Computer Society, Washington, DC, USA (2010)
14. Catmull, E., Clark, J.: Recursively generated b-spline surfaces on arbitrary topological meshes. Comput. Aided Des. **10**(6), 350–355 (1978)
15. Chen, X., Neubert, B., Xu, Y., Deussen, O., Kang, S.: Sketch-based tree modeling using markov random field. ACM Trans. Graph. **27**(5), 1–9 (2008)
16. Deussen, O., Lintermann, B.: Digital Design of Nature: Computer Generated Plants and Organs. Springer, New York, NY, USA (2005)
17. Doo, D., Sabin, M.: Behavior of recursive division surfaces near extraodinary points. Comput. Aided Des. **10**(6), 356–360 (1978)
18. Fabri, A., Giezeman, G.J., Kettner, L., Schirra, S., Schonherr, S.: On the design of cgal a computational geometry algorithms library. Software Pract. Exper. **30**(11), 1167–1202 (2000)
19. Galbraith, C., Mndermann, L., Wyvill, B.: Implicit visualization and inverse modeling of growing trees. In: Comput. Graph. Forum, vol. 23, pp. 351–360 (2004)
20. Gourmel, O., Barthe, L., Cani, M.P., Wyvill, B., Bernhardt, A., Paulin, M., Grasberger, H.: A gradient-based implicit blend. ACM Trans. Graph. **32**(2), 12:1–12:12 (2013)
21. Hart, J.C., Baker, B.: Implicit modeling of tree surfaces. Implicit Surfaces '96, pp. 143–152 (1996)
22. Hubert, E.: Convolution surfaces based on polygons for infinite and compact support kernels. Graph. Models **74**(1), 1–13 (2012)
23. Hubert, E., Cani, M.P.: Convolution surfaces based on polygonal curve skeletons. J. Symb. Comput. **47**(6), 680 – 699 (2012)
24. Ji, Z., Liu, L., Wang, Y.: B-mesh: a modeling system for base meshes of 3d articulated shapes. Comput. Graph. Forum **29**(7), 2169–2178 (2010)
25. Jin, X., Tai, C.: Analytical methods for polynomial weighted convolution surfaces with various kernels. Comput. Graph. **26**(3), 437–447 (2002)
26. Jin, X., Tai, C.: Convolution surfaces for arcs and quadratic curves with a varying kernel. Vis. Comput. **18**(8), 530–546 (2002)
27. Jin, X., Tai, C., Zhang, H.: Implicit modeling from polygon soup using convolution. Vis. Comput. **25**(3), 279–288 (2009)
28. Kettner, L.: $\sqrt{3}$ subdivision. In: ACM SIGGRAPH 2000, SIGGRAPH'00, pp. 103–112. ACM, New York, NY, USA (2000)
29. Lin, J., Jin, X., Wang, C.C.: Fusion of disconnected mesh components with branching shape. Vis. Comput. **26**(6-8), 1017–1025 (2010)
30. Lindenmayer, A.: Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. J. Theor. Biol. **18**(3), 300–315 (1968)
31. Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H., El-sana, J.: Automatic reconstruction of tree skeletal structures from point clouds. In: ACM SIGGRAPH Asia 2010, SIGGRAPH ASIA'10, pp. 151:1–151:8. ACM, New York, NY, USA (2010)
32. Lluch, J., Viv, R., Monserrat, C.: Modelling tree structures using a single polygonal mesh. Graph. Models **66**(2), 89–101 (2004)
33. Longay, S., Runions, A., Boudon, F., Prusinkiewicz, P.: Treesketch: interactive procedural modeling of trees on a tablet. In: Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling, SBIM '12, pp. 107–120. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2012)
34. Loop, C.: Smooth subdivision surfaces based on triangles. Master's Thesis, Depertent of Mathematics, University of Utah (1987)
35. Loop, C., Schaefer, S., Ni, T., Castano, I.: Approximating subdivision surfaces with gregory patches for hardware tesselation. ACM Trans. Graph. **28**(5), 1–9 (2009)
36. Maréchal, N., Galin, E., Guérin, E., Akkouche, S.: Component-based model synthesis for low polygonal models. In: Proceedings of Graphics Interface 2010, GI '10, pp. 217–224. Canadian Information Processing Society, Toronto, Ont., Canada, Canada (2010)
37. McCormack, J., Sherstyuk, A.: Creating and rendering convolution surfaces. Comput. Graph. Forum **17**(2), 113–120 (1998)
38. Neubert, B., Franken, T., Deussen, O.: Approximate image-based tree-modeling using particle flows. ACM Trans. Graph. **26**(3), 88–95 (2007)

39. Okabe, M., Owada, S., Igarashi, T.: Ineractive design of botanical trees using freehand sketches and example-based editing. Comput. Graph. Forum **24**(3), 487–496 (2005)
40. Overveld, K.v., Wyvill, B.: Shrinkwrap: an efficient adaptive algorithm for triangulating an iso-surface . Vis. Comput. **20**(6), 362–369 (2004)
41. Palubicki, W., Horel, K., Longay, S., Runions, A., Lane, B., Mech, R., Prusinkiewicz, P.: Self-organizing tree models for image synthesis. In: ACM SIGGRAPH 2009, SIGGRAPH'09, pp. 1–10. ACM, New York, NY, USA (2009)
42. Patney, A., Ebeida, M.S., Owens, J.D.: Parallel view-dependent tessellation of catmull-clark subdivision surfaces. In: Proceedings of the Conference on High Performance Graphics 2009, HPG'09, pp. 99–108. ACM, New York, NY, USA (2009)
43. Pirk, S., Stava, O., Kratt, J., Said, M.A.M., Neubert, B., Měch, R., Benes, B., Deussen, O.: Plastic trees: interactive self-adapting botanical tree models. ACM Trans. Graph. **31**(4), 50:1–50:10 (2012)
44. Reche-Martinez, A., Martin, I., Drettakis, G.: Volumetric reconstruction and interactive rendering of trees from photographs. ACM Trans. Graph. **23**(3), 720–727 (2004)
45. Schmidt, R., Singh, K.: meshmixer: an interface for rapid mesh composition. In: ACM SIGGRAPH 2010 Talks, SIGGRAPH'10, pp. 6:1–6:1. ACM, New York, NY, USA (2010)
46. Schwarz, M., Stamminger, M.: Fast gpu-based adaptive tessellation with cuda. Comput. Graph. Forum **28**(2), 365–374 (2009)
47. Sherstyuk, A.: Kernel functions in convolution surfaces: a comparative analysis. Vis. Comput. **15**(4), 171–182 (1999)
48. Sovakar, A., Kobbelt, L.: Api design for adaptive subdivision schemes. Comput. Graph. **28**(1), 67–72 (2004)
49. Tai, C., Zhang, H., Fong, C.: Prototype modeling from sketched silhouettes based on convolution surfaces. Comput. Graph. Forum **23**(4), 71–83 (2004)
50. Talton, J.O., Lou, Y., Lesser, S., Duke, J., Mech, R., Koltun, V.: Metropolis procedural modeling. ACM Trans. Graph. **30**(2), 11:1–11:14 (2011)
51. Tan, P., Fang, T., Xiao, J., Zhao, P., Quan, L.: Single image tree modeling. ACM Trans. Graph. **27**(5), 1–7 (2008)
52. Tan, P., Zeng, G., Wang, J., Kang, S.B., Quan, L.: Image-based tree modeling. In: ACM SIGGRAPH 2007, SIGGRAPH'07, pp. 87–93. ACM, New York, NY, USA (2007)
53. Vaillant, R., Barthe, L., Guennebaud, G., Cani, M.P., Rohmer, D., Wyvill, B., Gourmel, O., Paulin, M.: Implicit skinning: real-time skin deformation with contact modeling. ACM Trans. Graph. **32**(4), 125:1–125:12 (2013)
54. Wither, J., Boudon, F., Cani, M.P., Godin, C.: Structure from silhouettes: a new paradigm for fast sketch-based design of trees. Comput. Graph. Forum **28**(2), 541–550 (2009)
55. Wyvill, G., McPheeters, C., Wyvill, B.: Data Structure for Soft Objects. Vis. Comput. **2**(4), 227–234 (1986)
56. Xia, J., Garcia, I., He, Y., Xin, S.Q., Patow, G.: Editable polycube map for gpu-based subdivision surfaces. In: Symposium on Interactive 3D Graphics and Games 2011, I3D'11, pp. 151–158. ACM, New York, NY, USA (2011)
57. Xu, H., Gossett, N., Chen, B.: Knowledge and heuristic-based modeling of laser-scanned trees. ACM Trans. Graph. **26**(4), 19–31 (2007)
58. Zanni, C., Bernhardt, A., Quiblier, M., Cani, M.P.: Scale-invariant integral surfaces. Comput. Graph. Forum DOI 10.1111/cgf.12199. URL http://dx.doi.org/10.1111/cgf.12199
59. Zhu, X., Guo, X., Jin, X.: Efficient polygonization of tree trunks modeled by convolution surfaces. SCI. CHINA Ser. F **56**(3), 1–12 (2013)
60. Zhu, X., Jin, X., Liu, S., Zhao, H.: Analytical solutions for sketch-based convolution surface modeling on the gpu. Vis. Comput. **28**(11), 1115–1125 (2012)